# Architect – Tech to know

[**Awesome Software Architecture**](https://awesome-architecture.com/)

* [mehdihadeli](https://github.com/mehdihadeli)/[awesome-software-architecture](https://github.com/mehdihadeli/awesome-software-architecture)

[Beyond Senior Engineer: How to Think Like an Architect](https://medium.com/@admilavec/beyond-senior-engineer-how-to-think-like-an-architect-15f69ca7057b)

[Beyond Senior Engineer: Architecture Patterns](https://medium.com/@admilavec/beyond-senior-engineer-part-two-1f70c39ab383)

# Design Principles and Patterns

## Software Design Principles

* [Common Design principles](https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles#common-design-principles)
  + [Separation of Concerns](https://en.wikipedia.org/wiki/Separation_of_concerns) (SoC) -- Layered Architecture
  + [Keep It Simple Stupid](https://thevaluable.dev/kiss-principle-explained/) (KISS)
  + Don’t Repeat Yourself (DRY)
  + [Tell, Don’t Ask](https://deviq.com/tell-dont-ask/)
  + [You Ain’t Gonna Need It](https://martinfowler.com/bliki/Yagni.html) (YAGNI )
  + Encapsulate What may Change
  + Composition over inheritance
  + Don't Call Us, We'll Call You
  + Delegation
  + Minimize upfront design
  + Principle of Least Knowledge
* SOLID – OOPs Principles
  + single responsibility principle
  + open closed principle
  + Liskov's substitution principle
  + interface segregation principle
  + dependency inversion principle

GRASP (object-oriented design) principles (or challenges)

* + Low coupling, high cohesion and seven other important rules
    - controller, creator, indirection, information expert, polymorphism, protected variations, and pure fabrication

## Design patterns

[Design Patterns — Part 1 (Object Oriented Programming)](https://medium.com/@mlengineer/design-patterns-part-1-object-oriented-programming-a76c835f2436)

* GOF design patterns

# Design and Software Architecture

**Backend Application development**

* (Modular) Monolith
* Microservices
  + Cloud Native development with Microservices
  + https://www.twilio.com/blog/polly-fallbacks-dot-net-service-communication
  + https://code-maze.com/creating-resilient-microservices-in-net-with-polly/
* Serverless
* Polyglot persistence
* Clean Arch and DDD
* Distributed or Event-driven.
  + RabbitMQ - Erlang

**Frontend Application development**

* ([Micro-FE](https://scriptedalchemy.medium.com/micro-fe-architecture-webpack-5-module-federation-and-custom-startup-code-9cb3fcd066c))
  + [Webpack 5, Module Federation, and custom startup code](https://scriptedalchemy.medium.com/micro-fe-architecture-webpack-5-module-federation-and-custom-startup-code-9cb3fcd066c)
  + [Communication Between Micro-front ends](https://itnext.io/communication-between-micro-front-ends-f3c04d4c138e)

## Architectural Styles and patterns

[Difference between Architectural Style, Architectural Patterns and Design Patterns](https://www.geeksforgeeks.org/difference-between-architectural-style-architectural-patterns-and-design-patterns/)

[Architectural Styles vs. Architectural Patterns vs. Design Patterns](https://herbertograca.com/2017/07/28/architectural-styles-vs-architectural-patterns-vs-design-patterns/)

**Architectural Style** is abstract i.e. conceptual neither technology nor library.

An **Architectural Pattern** is concrete i.e. implementation of an Architectural Style.

**Structure architectural styles:** such as layered, pipes and filters and component-based styles.

**Messaging styles:** such as Implicit invocation, asynchronous messaging and publish-subscribe style.

**Distributed systems:** such as service-oriented, peer to peer style, object request broker, and cloud computing styles.

**Shared memory styles:** such as role-based, blackboard, database-centric styles.

**Adaptive system styles:** such as microkernel style, reflection, domain-specific language styles.

Examples of architectural patterns are microservices, message bus, service requester/ consumer, MVC, MVVM, microkernel, n-tier, domain-driven design, and presentation-abstraction-control

* *Layered – clean arch*
  + [*Layers in DDD microservices*](https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice)
* *SOA*
* *Event-based* architecture
* Microservices
* Etc.

Links:

[Software Engineering | Software Design Process - GeeksforGeeks](https://www.geeksforgeeks.org/software-engineering-software-design-process/?ref=rp)

[Software Engineering | Architectural Design - GeeksforGeeks](https://www.geeksforgeeks.org/software-engineering-architectural-design/?ref=rp)

[Difference Between Architectural Style, Architectural Patterns and Design Patterns](https://www.geeksforgeeks.org/difference-between-architectural-style-architectural-patterns-and-design-patterns/)

[Difference Between Technical Design and Conceptual Design in Software Engineering](https://www.geeksforgeeks.org/difference-between-technical-design-and-conceptual-design-in-software-engineering/?ref=rp)

## Presenting the Design and architecture

Software design aims to help developers transform requirements into implementation

[Architecture vs Design: The Difference Explained](https://simplicable.com/new/architecture-vs-design)

[Software Design vs. Software Architecture [closed]](https://stackoverflow.com/questions/704855/software-design-vs-software-architecture)

[What’s the difference between software architecture and design?](https://medium.com/@concisesoftware/whats-the-difference-between-software-architecture-and-design-b705c2584631)

## High-level design

* High Level Architecture
  + System Design
* Detailed-Level Design
  + Component-Level Diagrams

## Requirements

### Non-functional Requirements (NFRs)

[PASSME](https://jfdeclercq.biz/en/2016/06/02/passme/)

# Basics or Fundamentals

## Computer Network fundamentals

[Computer Network Tutorials](https://www.geeksforgeeks.org/computer-network-tutorials/?ref=lbp)

[Network Topology: 6 Network Topologies Explained & Compared](https://www.comparitech.com/net-admin/network-topologies-advantages-disadvantages/)

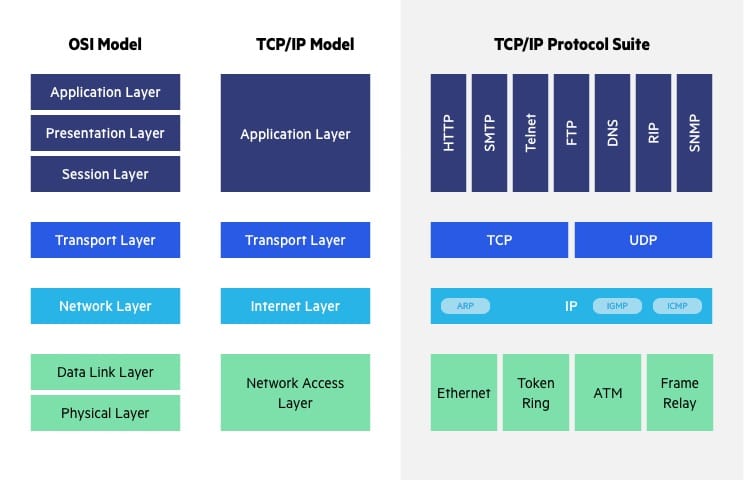
[Grokking Computer Networking for Software Engineers - Learn Interactively (educative.io)](https://www.educative.io/courses/grokking-computer-networking?affiliate_id=5073518643380224)

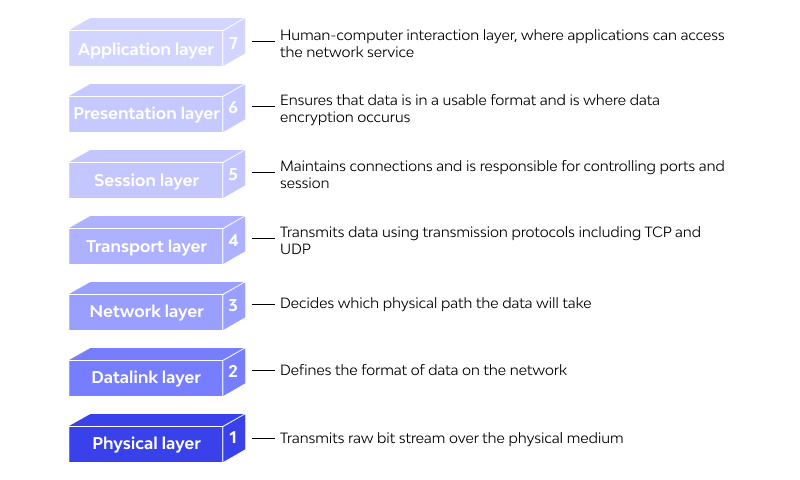
[7 Best Books and Courses to Learn Computer Networking, TCP/IP and UDP Protocols | by javinpaul | Javarevisited | Medium](https://medium.com/javarevisited/5-best-books-and-courses-to-learn-computer-networking-tcp-ip-and-udp-protocols-5a0e4dce75fa)

## [Internet protocol suite](https://en.wikipedia.org/wiki/Internet_protocol_suite) - [TCP/IP](https://www.imperva.com/learn/ddos/tcp-transmission-control-protocol/) | [OSI Model](https://www.imperva.com/learn/application-security/osi-model/)

The modern Internet is not based on OSI, but on the simpler TCP/IP model. TCP/IP specify how [data is exchanged online](https://www.imperva.com/learn/data-security/structured-and-unstructured-data/)

* [Application layer Protocols](https://www.geeksforgeeks.org/protocols-application-layer/) – HTTP, SOAP, MMQT, AMQP, SMTP etc.
* Transport layer Protocols – TCP, UDP, QUIC etc.
* Internet layer Protocols – IP(v4, v6) etc.
* Network Access/Link layer Protocols





**Links**

[TCP/IP Model - GeeksforGeeks](https://www.geeksforgeeks.org/tcp-ip-model/)

* [Comparing TLS and DTLS | Baeldung on Computer Science](https://www.baeldung.com/cs/tls-vs-dtls)
* [A brief overview of the TCP/IP model, SSL/TLS/HTTPS protocols and SSL certificates | by Uday Hiwarale | JsPoint | Medium](https://medium.com/jspoint/a-brief-overview-of-the-tcp-ip-model-ssl-tls-https-protocols-and-ssl-certificates-d5a6269fe29e)
* [Learning the TCP/IP Protocol Suite | by Vikas Yadav | codeburst](https://codeburst.io/learning-tcp-ip-protocol-suite-6947b601ea11)
* [TCP/IP Is Eating The World. Allow me to be blunt. TCP/IP is eating… | by Daniel Mezick | Medium](https://medium.com/@DanielMezick/tcp-ip-is-eating-the-world-fd8ba726e4cc)

[OSI model](https://www.wallarm.com/what/osi-model-explanation) - Open Systems Interconnection (OSI)

* [Breaking down the OSI model by buying pizza | by Ross M | Medium](https://medium.com/@rossim/breaking-down-the-osi-model-by-buying-pizza-8f6fc2e54319)
* [The OSI Model Explained: Handy Mnemonics to Memorize the 7 Layers (comparitech.com)](https://www.comparitech.com/net-admin/osi-model-explained/)
* [***Remembering the OSI layers with pizza and alligators 🍕🐊 | by Kyle Shelton | Medium***](https://medium.com/@chaoskyle/remembering-the-osi-layers-with-pizza-and-alligators-bf900e58d855)
  + Please Do Not Touch Steves Pet Alligator (Mnemonic for layers)

**OSI Layers Mnemonic**

* Layer 1: Physical = Please.
* Layer 2: Data Link = Do.
* Layer 3: Network = Not.
* Layer 4: Transport = Touch.
* Layer 5: Session = Steve's.
* Layer 6: Presentation = Pet.
* Layer 7: Application = Alligator.

**Others:**

Constrained Application Protocol (CoAP)

Extensible Messaging and Presence Protocol (XMPP)

### Communications [Protocols](https://github.com/stn1slv/awesome-integration#protocols) types

* Text protocol
  + HTTP, SOAP other e.g. [SMTP](https://www.rfc-editor.org/rfc/rfc5321), [HTTP](https://www.rfc-editor.org/rfc/rfc2616), [SIP](https://www.rfc-editor.org/rfc/rfc3261)
* Binary protocol
  + [MessagePack](https://msgpack.org/index.html), other e.g.  [RTP](https://www.rfc-editor.org/rfc/rfc3550), [TCP](https://www.rfc-editor.org/rfc/rfc793), [IP](https://www.rfc-editor.org/rfc/rfc791)
  + [binary protocols v. text protocols](https://stackoverflow.com/questions/2645009/binary-protocols-v-text-protocols/2645168#2645168) | [binary vs text](https://gist.github.com/samoshkin/18e8001715f15cc01f950df835a9d713)

### [Messaging protocol](https://developer.ibm.com/articles/messaging-protocols/)

* [MQTT](https://www.google.com/search?rlz=1C1CHBF_enIN871IN871&q=MQTT+is+protocol&spell=1&sa=X&ved=2ahUKEwj70P2Z4L33AhW24XMBHbPTDYkQirwEKAB6BAgCEDI) - [Message Queue Telemetry Transport Protocol](https://www.codeproject.com/Articles/5283088/MQTT-Message-Queue-Telemetry-Transport-Protocol-wi)
  + MQTT is payload agnostic, We can use a simple byte array, a simple string or a JSON
* **AMQP** Advanced Message Queuing Protocol is an open standard for passing business messages between applications or organizations
* [STOMP](http://stomp.github.io/) - Simple (or Streaming) Text Orientated Messaging Protocol

## Remote Procedure Call (RPC)

* RPC APIs e.g. SignalR provides an API for creating server-to-client [RPC](https://wikipedia.org/wiki/Remote_procedure_call), RabbitMQ provide RPC API for AMQP & STOMP,

## Messaging Systems

Message oriented middleware

### Message Format ([Data-serialization formats](https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats)) – [C# Serialization](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/)

*2 types of* [*communication protocols*](https://en.wikipedia.org/wiki/Communication_protocol) *text and binary*

* Text - human-readable data-interchange format
  + JSON
  + XML
    - and XML serialization
* Binary - binary data-interchange format
  + [Protobuf](https://dev.to/aspecto/is-protobuf-js-faster-than-json-ed4) -  IDL (Interface Definition Language) for gRPC
    - [Just like JSON but faster](https://dev.to/aspecto/is-protobuf-js-faster-than-json-ed4)
  + JSON 🡪 [BSON](https://en.wikipedia.org/wiki/BSON), [Smile](https://en.wikipedia.org/wiki/Smile_(data_interchange_format)), [UBJSON](https://en.wikipedia.org/wiki/UBJSON)
  + MessagePack

## Real-time web

* Polling
  + [~~Short polling~~](https://anuradha.hashnode.dev/short-polling-vs-long-polling-vs-web-sockets) ~~(a.k.a. AJAX based timer)~~ |[Non-blocking polling](https://access.redhat.com/documentation/en-us/red_hat_jboss_fuse/6.3/html/apache_cxf_development_guide/JAXWSAsyncDevPolling#JAXWSAsyncDevPollingNonBlock) - Periodical
  + Long polling (a.k.a. Comet based on XHR) | [Blocking polling](https://access.redhat.com/documentation/en-us/red_hat_jboss_fuse/6.3/html/apache_cxf_development_guide/JAXWSAsyncDevPolling#JAXWSAsyncDevPollingBlock)
* [*Server-Sent Events (SSE)*](https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events)
* [*WebSockets*](https://developer.mozilla.org/en-US/docs/Web/API/WebSocket)
* Webhooks pattern
  + User Defined HTTP Callback or User Defined Post HTTP Callback
* WebSub

# Development Approaches

## Test-driven development

## Domain-Driven Design

### Strategic Design

[DDD Strategic Patterns: How to Define Bounded Contexts - DZone](https://dzone.com/articles/ddd-strategic-patterns-how-to-define-bounded-conte)

[DDD Part 1: Strategic Domain-Driven Design | Vaadin](https://vaadin.com/blog/ddd-part-1-strategic-domain-driven-design)

### Tactical Design

### DDD Patterns

[DDD Concepts and Patterns - Introduction and Overview | Opus Software AG](https://opus.ch/ddd-concepts-and-patterns-introduction-and-overview/)

# Backend Services

[What Is Backend Web Architecture? Elements, Types and Benefits](https://www.indeed.com/career-advice/career-development/what-is-backend-web-architecture)

[**How to Become a Good Backend Engineer (Fundamentals)**](https://medium.com/@hnasr/how-to-become-a-good-backend-engineer-fundamentals-4dcc4a16ce55)

**Almost all the protocols we use and love on the backend are built on top of either TCP or UDP**.

**Communication Protocols**

Sometimes the backend architecture requires real-time Bidirectional communication protocols to build chatting, gaming apps or just communicating between two services. Protocols such as **WebSockets**, **gRPC** or just raw **TCP**/**UDP** can be used.

**TCP**

* TCP is a stream-based connection-oriented protocol while UDP is a message based and connectionless
* HTTP/2 - **HTTP protocol** was originally built on top TCP

**UDP**

* while UDP is a message based and connectionless
* HTTP/3 - HTTP/2 evolved and had to be rewritten on top of UDP with HTTP/3

## RPC

## Specifications

### Open API

* + Web API or **REST** and SOAP API
    - HATEOAS
      * OData Specification
      * JSON-API Specification
  + 4 Maturity Levels of REST API Design
    - [Richardson Maturity Model](https://restfulapi.net/richardson-maturity-model/)

[REST](https://apisyouwonthate.com/blog/understanding-rpc-rest-and-graphql) – RPC, REST and GraphQL

* REST is architectural style for distributed hypermedia systems.
* Constraints of REST architecture
* These representations portray data from various sources as simple "resources", or "collections" of resources, which are then potentially modifiable with actions and relationships being made discoverable via a concept known as hypermedia controls (HATEOAS)
* To fit REST into AMQP you would need to define hypermedia controls somehow (potentially an array of messages you could call next)
* [RPC Framework VS REST | The Detailed Guide](https://www.xenonstack.com/insights/rpc-framework-vs-rest)

### JSON:API Specification

### **GraphQL**

* GraphQL is a query language that lets you query an API for specific data through a single endpoint
  + [GraphQL and REST Level 3 (HATEOAS)](https://techblog.commercetools.com/graphql-and-rest-level-3-hateoas-70904ff1f9cf)
  + [HATEOAS vs GraphQL decision criteria set for microservices? [closed]](https://stackoverflow.com/questions/46061755/hateoas-vs-graphql-decision-criteria-set-for-microservices)

### gRPC

[Interface Definition Language - gRPC for WCF Developers | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/architecture/grpc-for-wcf-developers/interface-definition-language)

**What is faster than Protobuf?**

[Cap'n Proto](https://capnproto.org/) calls this “packing” the message; it achieves similar (better, even) message sizes to protobuf encoding, and it's still faster

# .Net

* Background task
* Cache
* RabbitMQ or Servicebus
* Performance - APM
* Logging and Exception handling
  + Serilog, ELK
* [Exception Handling](https://docs.abp.io/en/abp/latest/Exception-Handling#exception-handling)
  + [Consistent error responses](https://medium.com/swlh/clean-architecture-best-exception-handling-with-consistent-responses-in-asp-net-core-api-b22b07a08e38)
    - Error Message Format

# UI/UX

* Angular/React
* Canvas
* Micro-Frontend
* SEO and web site Analytics
  + Clarity (+ Google Analytics)
    - Google Analytic

Extending

* Create reusable components and publish in Verdaccio – npm packages
* Web Components

# Mobile And Cross Platform

* Xamarin
* Kotlin
* Flutter

# Cloud

* Azure
* AWS
* GCP

# Security

* Cyber Security
* OWASP Top 10
* CWE/SANS TOP 25

# Data

* Data integrations
* Data Science and Big Data Analysis
  + [Can .Net Developers Do Big Data Analysis / Data Science?](https://medium.com/tunapanda-institute/can-net-developers-do-big-data-analysis-data-science-3d8924b29450)
  + https://www.datasciencecentral.com/learning-path-to-become-a-data-scientist-for-asp-net-developers/
  + [Advantages of Big Data Analytics and Data Science Integration with ASP.NET](https://www.datasciencecentral.com/advantages-of-big-data-analytics-and-data-science-integration/)
  + [Can .Net Developers Do Big Data Analysis / Data Science?](https://medium.com/tunapanda-institute/can-net-developers-do-big-data-analysis-data-science-3d8924b29450)
* Business intelligence and Data analytics
  + Both roles are relatively similar in definition, process, type of data, and type of analysis, except for the type of tools used, which may vary slightly
  + [Continues Intelligence](https://www.xenonstack.com/insights/continuous-intelligence)
* Data fabric
* Data Agnostic - ETL
  + Apache Beam vs Spark

## Database: Multi-model and multi-purpose Database

* **PostgreSQL** (Multi-Model)
* **Azure CosmosDB** (Multi-Model, Multi-purpose)
* **SingleStore** (OLAP and OLTP)

**Traditional DB types**

* **RDBMS** for Transactional use cases with structured data
* **Wide-Column Database** for low-latency, distributed Database
* **Key-Value Store** for Distributed Cache
* **Graph Database** for extremely relational data
* **Document Database** for transactional use cases with semi-structured data
* **Search Engines** for full-text and advanced search (e.g., location-based search).
* **Distributed SQL** for low-latency, distributed database with a transactional guarantee.
* **OLAP Database** for Data-warehouse and data analytics

## Data Processing Architecture

[Lambda or Kappa? The need for a new data processing architecture](https://www.qlik.com/blog/lambda-or-kappa-the-need-for-a-new-data-processing-architecture)

# Integrations

Legacy system integration

Enterprise application integration (EAI)

Third-party system integration

Business-to-business integration

[System integration component](https://www.slideteam.net/system-integration-components-example-ppt-presentation.html)

* Business Processes
* Internal Apps
* ERP
* CRM
* Database
* Legacy system

[How to Choose Between Different Integration Approaches.pdf](https://www.actian.com/wp-content/uploads/2020/01/Approach-to-Integration-eBook.pdf)

* Manual data integration
* Middleware data integration
* Application-based integration
* Uniform access integration
* Common storage integration

[Objects and methods of integration of systems](https://flexberry.github.io/en/gbt_integration-methods.html) | Approaches

1. Integration platforms
   1. Remote procedure call (RPC, Web-services, REST, etc.)
2. Middleware (Microsoft.Net, Java Runtime)
3. [Data Virtualization](https://www.altexsoft.com/blog/data-virtualization/) | [**Data Virtualization 101**](https://www.g2.com/articles/data-virtualization)
   1. Data federation is kind of Data virtualization
4. Data integration (DI)
   1. Universal data access
   2. Data warehouse
5. Application integration | Enterprise Application Integration (EAI)
   1. Application programming interfaces
   2. Messaging (Enterprise service bus)
   3. Service-oriented architecture
   4. Integration of user interfaces
6. Integration of business processes
7. Electronic Document Integration/Interchange (EDI)

[Application Integration vs. Data Integration](https://www.ibm.com/cloud/blog/application-integration-vs-data-integration)

* **Data Integration** deals with large sets of data at rest; it happens when the process that created the data has been completed.
* **Application Integration**, on the other hand, is for integrating real-time data between two or more applications

[Event-Driven Architecture as a Strategy](https://dzone.com/articles/event-driven-architecture-as-a-strategy)

## Application Integration - Software/system Integration

[How to Choose Between Different Integration Approaches](https://www.actian.com/wp-content/uploads/2020/01/Approach-to-Integration-eBook.pdf)

**Application integration stages**

Scoping & Planning 🡪 Integrated System Design 🡪 Tech Stack Selection, Evaluation and Comparison 🡪 implementation and Testing

* [**Protocols**](https://github.com/stn1slv/awesome-integration#protocols) - REST, SOAP, ESB, etc.
* [App integration approaches](https://www.scnsoft.com/application/integration) | [Types of application integration](https://www.actian.com/wp-content/uploads/2020/01/Approach-to-Integration-eBook.pdf)
  + Direct (point-to-point) integration
    - [When To Use Point-To-Point Integration](https://www.itconvergence.com/blog/when-to-use-point-to-point-integration/)
  + Mediated integration | middleware-based integration
    - Enterprise Service Bus (ESB)
    - iPaaS
  + Using specific software (‘robots’)
  + [Models of Enterprise Application Integration (EAI)](https://www.rishabhsoft.com/blog/enterprise-application-integration)
    - Point-to-Point Integration |
    - Hub-and-Spoke Integration
    - Bus Integration
    - Middleware-Based
    - Microservices
* Application integration Strategy
  + Batch Data Integration
  + Shared Database | Common data storage
  + Remote Procedure Invocation
  + Messaging Strategy
  + SOA/ESB
* Application integration architecture
* best practices
* B2B partnerships

[App Integration: Building Apps that Interface with Others](https://www.inspiringapps.com/Blog/47/app-integration-building-apps-that-interface-with-others/)

[Application Integration: Steps, Skills, Success Factors](https://www.scnsoft.com/application/integration)

[How to build an application integration framework for flexibility](https://www.theserverside.com/feature/How-to-build-an-application-framework-for-flexibility)

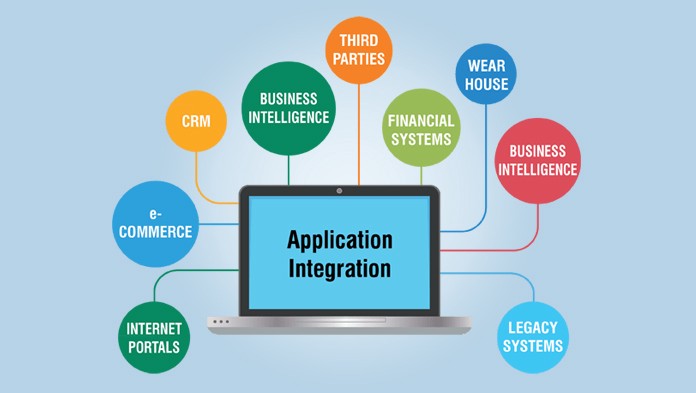
[Enterprise Application Integration: How To Digitally Transform Your Business](https://www.rishabhsoft.com/blog/enterprise-application-integration)

[Approaching Application Integration](https://www.informit.com/articles/article.aspx?p=169480&seqNum=2)

[Enterprise Integration Patterns Books](https://www.enterpriseintegrationpatterns.com/books1.html)

[Application Integration: A Detailed Primer](https://www.jitterbit.com/blog/what-is-application-integration/)

[What is Application Integration? Definition and Benefits](https://www.astera.com/type/blog/application-integration/)



## Data Integration

[Types of Data Integration: ETL vs ELT and Batch vs Real-Time](https://www.striim.com/blog/data-integration/)

ETL VS ELT

* Batch data integration - [Non-Real-time](https://www.geeksforgeeks.org/difference-between-real-time-tasks-and-non-real-time-tasks/)
  + [Batch job](https://www.quora.com/What-are-the-advantages-and-disadvantages-of-batch-processing) | [batch processing](https://www.itrelease.com/2012/12/what-are-advantages-and-disadvantages-of-batch-processing-systems/)
  + Recurring or repetitive
  + C# Batch jobs – Background tasks and workers - [BackgroundService](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-6.0&tabs=visual-studio#timed-background-tasks)
    - [Schedule Jobs](https://github.com/thangchung/awesome-dotnet-core#scheduler-and-job) - [Hangfire](https://www.hangfire.io/), Quartz.NET , [Cron jobs](https://codeburst.io/schedule-cron-jobs-using-hostedservice-in-asp-net-core-e17c47ba06), [Vanlightly](https://github.com/Vanlightly)/[Taskling.NET](https://github.com/Vanlightly/Taskling.NET)
    - [Process your list in parallel to make it faster in .NET](https://timdeschryver.dev/blog/process-your-list-in-parallel-to-make-it-faster-in-dotnet)
* [Real-time data integration](https://www.striim.com/real-time-data-integration/) with [change data capture](https://www.striim.com/blog/change-data-capture-cdc-what-it-is-and-how-it-workshttps:/www.striim.com/change-data-capture/)
  + CDC
    - Audit Columns
    - Table Deltas
    - Trigger-based CDC
    - Log-Based Change Data Capture

#### ETL

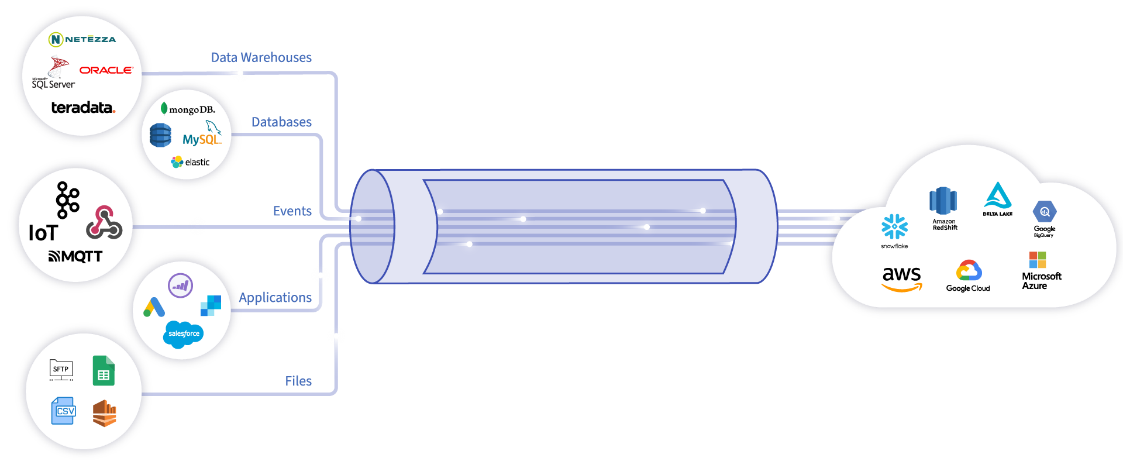
[SQL Server ETL](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/) -

Paid Tools:

* [Hevo Data](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#hevo)
* [Informatica PowerCenter](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#informatica)
* [Striim](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#striim)
* [Pentaho](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#pentaho)
* [IBM Infosphere DataStage](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#IBM)
* [Oracle GoldenGate](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#oracle)
* [Qlik Replicate](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#qlik)

Free Tools

* [SSIS](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#Microsoft_SQL)
* [Talend Open Studio](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#talend)
* [Apache Nifi](https://hevodata.com/learn/microsoft-sql-server-etl-best-tools/#nifi)
* C# - [Fundamentals of Parallel Programming and ETL in C#](https://medium.com/swlh/parallel-etl-in-c-3394342e6e64)
  + [paillave](https://github.com/paillave)/[Etl.Net](https://github.com/paillave/Etl.Net) **and ETLBox (Commercial)**
  + [Cinchoo](https://github.com/Cinchoo)/[ChoETL](https://github.com/Cinchoo/ChoETL) **&** [madhon](https://github.com/madhon)/[ReactiveETL](https://github.com/madhon/ReactiveETL) **(Rhino ETL)**
  + [gridsum](https://github.com/gridsum)/[DataflowEx](https://github.com/gridsum/DataflowEx)
  + <https://github.com/topics/etl?l=c%23>
  + [JohnnyBravo75](https://github.com/JohnnyBravo75)/[DataBridge.NET](https://github.com/JohnnyBravo75/DataBridge.NET)



Links:

[Applied Architecture Patterns on the Microsoft Platform, 2nd Edition 2014](https://media.oiipdf.com/pdf/702fa290-cb80-48dd-beca-77e9fcd746b8.pdf)

#### Batch processing

# Real-Time Streaming

* Apache Flink

# No Code or Low Code

Rapid Application Development: Low Code/No Code (LCNC)

* Citizen development

In 2022 as well, we will see more and more use cases for LCNC:

* Web/Mobile App development
* Websites and landing pages
* Intelligent Chatbots using dialog flow
* E-commerce
* Machine Learning
* Artificial Intelligence (Video, Audio, Image)
* Workflow management
* Process Automation using RPA

Chatbots and Conversation As A Platform (CAAP)

[SciSharp](https://github.com/SciSharp)/**[BotSharp](https://github.com/SciSharp/BotSharp)**

# RPA - Automation with Software robots/bots (AI +ML)

* Blue Prism/AA (Automation Anywhere)/UiPath/OpenSpan (Pega)/WorkFusion/Robocorp
* https://www.analyticsinsight.net/top-10-open-source-free-rpa-tools-2020/
* Blue Prism (C#) - [Blue Prism Vs OpenSpan](https://mindmajix.com/blue-prism-vs-openspan) & [Robotic Process Automation vs Traditional Test Automation](https://www.qentelli.com/thought-leadership/insights/robotic-process-automation-vs-traditional-test-automation)

# AI - Artificial Intelligence

* Machine Learning
* Deep Learning (AI)
* [AIOps - Artificial Intelligence for IT Operations (AIOps)](https://sciencelogic.com/solutions/aiops)
* Augmented reality AI
  + [Augmented Intelligence](https://digitalreality.ieee.org/publications/what-is-augmented-intelligence) (also known as intelligence amplification, or IA)

[Artificial Intelligence Technology Trends That Matter for Business in 2022](https://mobidev.biz/blog/future-artificial-intelligence-technology-ai-trends)

[Top 10 Artificial Intelligence Trends in 2022](https://www.mygreatlearning.com/blog/top-artificial-intelligence-trends/)

# polyglot architecture

Polyglot software engineer

[FullStack vs PolyGlot](https://dev.to/slimdestro/fullstack-vs-polyglot-1jbi)

# Application Modernization

[How to Build a Roadmap to App Modernization](https://thenewstack.io/how-to-build-a-roadmap-to-app-modernization/)

[The State of Application Modernization, 2022](https://thenewstack.io/the-state-of-application-modernization-2022/)

## Microservices

* ***Decoupling & Componentization***: set of small services | small piece of functionality
  + Shared Nothing Architecture - Stateless systems
  + Agility | easily replaced and upgraded
  + reusability
  + Granular Scaling
* ***Business Capabilities***: specific end-to-end domain or business capability within a certain context boundary (bounded context) | SRP
* ***Autonomy***: **developed autonomously** and be **deployable independently**
  + are usually autonomously developed
  + are independently deployable
  + use messaging to communicate
  + each deliver a certain business capability.
* ***Products not Projects*** | Responsibility:
* [**Smart endpoints and dumb pipes**](https://medium.com/@nathankpeck/microservice-principles-smart-endpoints-and-dumb-pipes-5691d410700f)
* **Decentralized Governance**
  + Mixed Technology Stack | language agnostic
    - Technology-agnostic approach - no “one true tech” - No "one size fits all
  + No standardized pattern or any technology pattern i.e. using the right tool for the right job
* **Decentralized Data Management**
  + Polyglot Persistence
* **Infrastructure Automation** (CD & CI) | Devops |containerization --> containers (Docker)|
* **Design for failure** | **Fault Isolation** - reliable & resilient
* ***Design to Change*** | ***Evolutionary Design***
* protocol-aware heterogeneous interoperability

### Polyglot persistence

### Polyglot **Microservices**

Dapr - [1/3 Polyglot microservice development with Dapr – What are polyglot Microservices?](https://whiteduck.de/en/what-are-polyglot-microservices/)

### Event-Driven Microservices Architecture

## Containers

## Cloud Native Development

Journey of Adopting Cloud-Native Development

* https://www.weave.works/technologies/going-cloud-native-6-essential-things-you-need-to-know/
* https://www.infosys.com/services/open-source/insights/adopt-cloud-native.html

### Cloud database

cloud database that uses machine learning to automate database tuning, security, backups, updates, and other routine management tasks traditionally performed by DBAs

* Oracle Autonomous Database & Azure SQL Database

# Misc

[**38 Actions and Insights to Become a Better Software Architect**](https://blog.kai-niklas.de/posts/7-38-actions-and-insights-to-become-a-better-software-architect/)

[The strategic role of a Solution Architect](https://svitla.com/blog/the-strategic-role-of-a-solution-architect)

[The role, skills, and duties of a software architect](https://syndicode.com/blog/the-role-skills-and-duties-of-a-software-architect/)

[The Complete Guide to Becoming a Software Architect Course](https://www.youtube.com/playlist?list=PLcuOsgecRieNEPrc6REQhTWsLLL7PlY1Q)

[10 Bad Coding Habits You Need to Put an End to Right Now](https://dev.to/muthuannamalai12/10-bad-coding-habits-you-need-to-put-an-end-to-right-now-363b)

## Roles and Responsibility

[Who Is a Technical Architect? Job Role, Responsibilities, Skills Required and More](https://www.simplilearn.com/technical-architect-article)

* good communication and problem-solving skills,
* ability to assimilate information,
* updated technical know-how,
* and ability to think ahead

### Job responsibilities

* Meet clients and relevant stakeholders to discuss hardware and software requirements
* Develop bespoke solutions in collaboration with cross-functional teams
* Provide technical guidance during the design, build, test, and deploy phases of the SDLC
* Build strategies to deliver an exceptional user experience to the customers
* Establish and maintain customer relationships
* Take ownership of tasks such as product evaluation, early-phase project estimates, and buy vs. build decisions
* Create technical documentation, RFPs, and professional reports

### Preferred skills

* Excellent organizational, interpersonal, and leadership skills
* Experience with JSON, XML, .NET, database modelling & design
* Experience with ML domains, GCP and AWS technologies
* Experience with operational risk management, G suite, and enterprise software solutions
* Knowledge of the latest practices for software engineering and system security measures
* A strong attention to detail with good presentation skills
* Experience in delivering great customer service with desired software solutions

# [Web Application and Software Architecture 101](https://www.educative.io/courses/web-application-software-architecture-101)

## Introduction

* About This Course
* Significance of Software Architecture

## Different Tiers in Software Architecture

* Introduction
* Single-Tier Applications
* Two-Tier Applications
* Three-Tier Applications
* N-Tier Applications
* Different Tiers in Software Architecture Quiz

## Web Architecture

* What is Web Architecture?
* Client-Server Architecture
* Client
* Types of Clients
* Server
* Communication between the Client and the Server
* Web Architecture Quiz - Part 1
* What is a REST API?
* HTTP Push and Pull - Introduction
* HTTP Pull - Polling With AJAX
* HTTP Push
* HTTP Push-Based Technologies
* Client-Side vs. Server-Side Rendering
* Web Architecture Quiz - Part 2

## Scalability

* What is Scalability?
* Types of Scalability
* Which Scalability Approach is Right for our App?
* Primary Bottlenecks That Hurt the Scalability of our Application
* How to Improve and Test the Scalability of our Application?
* Scalability Quiz

## High Availability

* What is High Availability?
* Reasons for System Failures
* Achieving High Availability - Fault Tolerance
* Redundancy
* Replication
* High Availability Clustering
* High Availability Quiz

## Load Balancing

* Introduction to Load Balancing
* Understanding DNS – Part 1
* Understanding DNS – Part 2
* DNS Load Balancing
* Load Balancing Methods
* Load Balancing Quiz

## Monolith and Microservices

* What is Monolithic Architecture?
* When should you pick a Monolithic Architecture?
* What is Microservice Architecture?
* When should you pick Microservices Architecture?
* Monolith and Microservices– Understanding the Trade-Offs – Part 1
* Monolith and Microservices– Understanding the Trade-Offs – Part 2
* Monolith and Microservices Quiz

## Micro Frontends

* Introduction to Micro Frontends
* The Need for Micro Frontends
* Micro Frontends Integration

## Database

* Introduction and Types of Data
* Relational Databases
* When should you pick a relational database?
* NoSQL Databases - Introduction
* Features of NoSQL Databases
* When to pick a NoSQL Database?
* Is NoSQL More Performant Than SQL?
* Database Quiz - Part 1
* Polyglot Persistence
* Multi-Model Databases
* Eventual Consistency
* Strong Consistency
* CAP Theorem
* Database Quiz - Part 2
* Types of Databases
* Document-Oriented Database
* Graph Database
* Key-Value Database
* Time Series Database
* Wide-Column Database
* Database Quiz - Part 3

## Caching

* Introduction
* Do I Need A Cache?
* Reducing the Application Deployment Costs via Caching
* Caching Strategies
* Caching Quiz

## Message Queue

* Introduction to Message Queues
* Publish-Subscribe Model
* Point-to-Point Model
* Notification Systems and Real-Time Feeds with Message Queues
* Handling Concurrent Requests with Message Queues
* Message Queue Quiz

## Stream Processing

* Introduction
* Data Ingestion
* Different Ways of Ingesting Data and the Challenges Involved
* Data Ingestion Use Cases
* Data Pipelines
* Distributed Data Processing
* Lambda Architecture
* Kappa Architecture
* Stream Processing Quiz

## More on Architecture

* Event-Driven Architecture - Part 1
* Event-Driven Architecture - Part 2
* Webhooks
* Shared-Nothing Architecture
* Hexagonal Architecture
* More on Architecture Quiz – Part 1
* Peer-to-Peer Architecture – Part 1
* Peer-to-Peer Architecture – Part 2
* Decentralized Social Networks
* Federated Architecture
* More on Architecture Quiz – Part 2

## Picking the Right Technology

* How to Pick the Right Server-Side Technology?
* Key Things to Remember When Picking the Tech Stack
* Conclusion

## Case Studies

* A Web-Based Mapping Service Like Google Maps
* A Baseball Game Ticket Booking Web Portal

## Mobile Apps

* Introduction
* Before You Design Your Mobile App
* Responsive Interfaces
* Types of Mobile Apps – Part 1
* Types of Mobile Apps – Part 2
* Choosing Between a Native and a Hybrid App
* Progressive Web Apps
* Mobile Backend as a Service
* Zero to Application Architect
* Epilogue
* Changelog

# Architecture and Design

* [System Architecture](https://www.themetisfiles.com/2013/01/the-four-types-of-system-architectures/)
* Enterprise Architecture
* Solution Architecture
* Technical Architecture
* Application/Software Architecture
* Technology Architecture
* Infrastructure Architecture
  + Network Architecture
* Data Architecture
* Process Architecture

Application Architecture AKA software Architecture

[Difference between System Architecture and Software Architecture](https://pediaa.com/what-is-the-difference-between-system-architecture-and-software-architecture/)

Software architecture is a type of system architecture

Architecture Description Languages (ADL)

Process architecture

## Designing Different systems

[System Design Interview Course](https://medium.com/double-pointer/system-design-interview-course-31ddb8dfdafc)

[How to Design Software](https://medium.com/@jgefroh/list/how-to-design-software-03066fa9dcbf)

[codekarle](https://github.com/codekarle)/[**system-design**](https://github.com/codekarle/system-design)

### Designing CPQ Systems (Configure-Price-Quote)

* [Designing CPQ Systems (Configure-Price-Quote)](https://www.linkedin.com/pulse/designing-cpq-systems-andy-pieroux)

### Designing Social Network

* [LinkedIn (or similar professional social network)](https://bit.ly/3HEto81)
* [Instagram or a Similar App (Snapchat, Flickr, Picasa) Design](https://bit.ly/3pkNmhD)
* [Facebook Messenger, WhatsApp, Slack, Discord or a Similar Messaging Application’s Design](https://bit.ly/35cfWec)
* [Twitter or Facebook Feed](https://bit.ly/35v7TsI)

### Designing Ecommerce

* [Amazon/ Flipkart/ Ebay or Similar E-commerce Applications](https://bit.ly/33VqCgA)
* [How to design sales inventory Microservice based on event-driven architecture?](https://medium.com/deutsche-telekom-gurgaon/how-to-design-sales-inventory-microservice-based-on-event-driven-architecture-e6f67e3a2e91)

#### Designing Food ordering system

* Zomoto
* Swiggy

### Designing booking system

* [BookMyShow, Fandango (or similar movie ticketing application)](https://bit.ly/3hlHUXc)
* Bus, Train, or Flight
* Cabs
  + [Uber or a Similar App (Grab, Lyft or Ola) Design](https://bit.ly/33ZcQcY)
* Hotels and tourism
  + [Airbnb or Similar Online Marketplace for Lodging](https://bit.ly/3HxmqSc)
  + Oyo

### Designing Video Streaming system

* [Video Streaming Service e.g. Netflix or YouTube Design](https://bit.ly/3IqXVqN)
* [NETFLIX System design | software architecture for netflix](https://www.youtube.com/watch?v=psQzyFfsUGU)
  + [System Design Interview: Video Streaming Service e.g. Netflix or YouTube Design](https://medium.com/double-pointer/system-design-interview-video-streaming-service-e-g-netflix-or-youtube-design-adc2402e54a1)
  + [Design Youtube / Netflix](https://astikanand.github.io/techblogs/high-level-system-design/design-youtube-or-netflix)
  + [Netflix System Design | YouTube System Design | System Design Interview Question](https://www.youtube.com/watch?v=lYoSd2WCJTo)
  + [Mastering Chaos - A Netflix Guide to Microservices](https://www.youtube.com/watch?v=CZ3wIuvmHeM)
  + [What technology stack is Netflix built on?](https://www.quora.com/What-technology-stack-is-Netflix-built-on)
  + [github - Netflix](https://github.com/Netflix)
    - [datastaxdevs](https://github.com/datastaxdevs)/[**workshop-graphql-netflix**](https://github.com/datastaxdevs/workshop-graphql-netflix)
  + Netflix videos storage
    - [Netflix | Open Connect](https://openconnect.netflix.com/)
    - [MezzFS — Mounting object storage in Netflix’s media processing platform](https://netflixtechblog.com/mezzfs-mounting-object-storage-in-netflixs-media-processing-platform-cda01c446ba)

### Designing Payment systems

[Domain-driven design practice — Modelling the payments system](https://medium.com/airwallex-engineering/domain-driven-design-practice-modeling-payments-system-f7bc5cf64bb3)

Payment Gateway integration in Angular and Web API

### Designing Recommendation System

* [System Design Interview: Recommendation System Design (As Used By YouTube, Netflix etc.)](Recommendation%20System%20Design%20(As%20Used%20By%20YouTube,%20Netflix%20etc.))

### Designing Search Engine

* [Search Engine](https://bit.ly/3hmgmB5)

### Designing Point Of Sales, Production Line applications and IOT

### Designing Unified Payment Interface (UPI)

[Technical understanding of the Unified Payment Interface (UPI) — Part (1/2) | by Chirag Bhattad | Medium](https://medium.com/@chiragbhattad/technical-understanding-of-the-unified-payment-interface-upi-part-1-ce0f4b8c1fcb)

### Others

* [Dropbox or a Similar File Storage & Sharing Service (Google Drive/ OneDrive) Design](https://bit.ly/3MhkUam)

# Aspects

## Networking and IT infrastructure

[Computer Network fundamentals](#_Computer_Network_fundamentals) section

* DMZ
* DNS
* Load balancing
* Server – Clusters nodes
* Failover

Roles

* SRE team
  + Site reliability engineers (SRE), Systems administrators, systems engineers, and operations experts

## Performance

### Cache

### Scalability

### Load balancing

IT operation

Load balancing distributes a workload across multiple servers to improve performance

[Load balancing: system design interview concepts (4 of 9)](https://igotanoffer.com/blogs/tech/load-balancing-system-design-interview" \l ":~:text=Load%20balancing%20is%20the%20process,%2C%20user%20sessions%2C%20and%20caching.)

[Load Balancer in System Design](https://medium.com/must-know-computer-science/system-design-load-balancing-1c2e7675fc27)

[System Design — Load Balancing](https://medium.com/must-know-computer-science/system-design-load-balancing-1c2e7675fc27)

Software load balancer

* Dynamic load balancing algorithms
  + Least Connection First Scheduling
  + Weighted least connection
  + Weighted response time
  + Resource-based
* Static load balancing algorithms
  + Round robin Scheduling
  + Weighted round robin
  + IP hash

Hardware load balancer

* Layer4 Hardware Load Balancing
* Layer7 Hardware Load Balancing

IIS

* [Web Farm Framework 2.0 for IIS 7](https://docs.microsoft.com/en-us/iis/web-hosting/microsoft-web-farm-framework-20-for-iis-7/overview-of-the-web-farm-framework-20-for-iis)

## High availability (HA) and Disaster recovery

* Fault tolerance and Redundancy
* In addition, this is IT operations task. Or SREs

**Running mission critical systems**

* programs that must function continuously in order for a business or segment of a business to be successful
* [Don't mix IT operational and non-operational tasks](https://itmlinstitute.org/dont-mix-it-operational-and-non-operational-tasks/" \l ":~:text=Operational%20tasks%20are%20the%20mission,the%20daily%20and%20nightly%20production.)
* [Operational tasks](https://facilethings.com/blog/en/operations)

### Server Clusters

Server clustering, on the other hand, combines multiple servers to function as a single entity. Cluster is a group of servers that run as if it were a single entity

[Choosing between a farm and a cluster](https://www.ibm.com/docs/en/filenet-p8-platform/5.5.x?topic=available-choosing-between-farm-cluster)

### Failover cluster

* Redundancy
* Active-Passive Failover Cluster

**IIS Failover**

Achieving high-availability of Internet Information Services (IIS) servers is by using **Network Load Balancing (NLB)**

[Load Balancing vs. Failover Clustering](https://nirajrules.wordpress.com/2009/11/20/load-balancing-vs-failover-clustering/)

**Load Balancing** is all about improvising performance (scale) while **Failover Clustering** is improvising uptimes mitigating system failures

**Load Balancing** happening at web/application servers (stateless hopefully) and **failover clustering** at database servers (state full)

[Load Balancing vs Server Clustering: Understand the Difference](https://firewalltimes.com/load-balancing-vs-server-clustering/#:~:text=Load%20balancing%20distributes%20a%20workload,function%20as%20a%20single%20entity.)

[**Achieving High Availability and Scalability - ARR and NLB**](https://learn.microsoft.com/en-us/iis/extensions/configuring-application-request-routing-arr/achieving-high-availability-and-scalability-arr-and-nlb)

* Microsoft Application Request Routing (ARR)
* [Network Load Balancing](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc770558(v=ws.11))
* [IIS (webserver) cluster](https://www.informaticar.net/how-to-create-iis-webserver-cluster-highly-available-iis-for-exchange-services-arr/)

[Configuring IIS World Wide Web Publishing Service in a Windows Server failover cluster](https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/iis/www-authentication-authorization/configure-w3svc-wsfc)

Windows Server failover cluster (WSFC)

* [Install and configure Network Load Balancing](https://learn.microsoft.com/en-us/iis/web-hosting/configuring-servers-in-the-windows-web-platform/network-load-balancing)
* [Network Load Balancing](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc770558(v=ws.11))

**SQL Server Failover**

[Windows Server Failover Clustering with SQL Server](https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/windows-server-failover-clustering-wsfc-with-sql-server?view=sql-server-ver15)

* [What is Windows Server Failover Clustering (WSFC)?](https://www.techtarget.com/searchwindowsserver/definition/Windows-Server-failover-clustering)

Disaster recovery for Microsoft SQL Server

[How to plan for SQL Server Disaster Recovery?](https://www.nucleustechnologies.com/blog/sql-server-disaster-recovery-plan/)

* Taking Database Backups Regularly.
* Using Always-On and Basic Database Availability Groups.
* Log-Shipping.
* Fail-over Clustering.
* Database Replication.
* Restore Master Database in SQL Server.

### Transactional replication

Transactional replication is a SQL Server technology that is used to replicate changes between two databases. These changes can include database objects like tables (primary key is required), stored procedures, views, and so on, as well as data.

Transactional Replication

What is the difference between transactional replication and merge replication?

[SQL Server replication configuration: Peer to Peer and Merge Replication](https://www.sqlshack.com/sql-server-replication-configuration-peer-to-peer-and-merge-replication/)

## Large Scale Distributed Systems

# Roles and Responsibilities

## Technical Skills:

What .NET technologies are you most familiar with?

Can you explain your experience with .NET Core?

Have you worked with Azure DevOps and CI/CD pipelines?

How do you handle version control using Git?

## System Design and Architecture:

Can you walk us through your approach to designing a .NET solution architecture?

How do you ensure the scalability and maintainability of your .NET solutions?

Can you explain your experience with microservices and API design?

How do you handle cross-cutting concerns such as security, logging, and caching in your .NET solutions?

## Project Management:

Have you managed .NET projects before? How did you ensure the project was delivered on time and within budget?

How do you prioritize tasks and allocate resources in your .NET projects?

How do you handle risks and challenges in your .NET projects?

## Communication and Collaboration:

Can you provide an example of how you have collaborated with cross-functional teams in a .NET project?

How do you communicate technical concepts to non-technical stakeholders in your .NET projects?

Can you explain your experience with agile methodologies and how you apply them in your .NET projects?

## Security and Compliance:

How do you ensure the security and compliance of your .NET solutions?

Can you explain your experience with implementing authentication and authorization in .NET solutions?

How do you handle data protection and privacy requirements in your .NET solutions?

Performance Optimization:

How do you optimize the performance and scalability of your .NET solutions?

Can you provide an example of how you have improved the performance of a .NET solution?

How do you handle caching and load balancing in your .NET solutions?